AMENDMENTS TO THE CLAIMS

1. (Currently Amended) In an electronic device, a method for management of software, comprising ~~the steps of~~:

providing an organizational structure in a storage having a plurality of members, each member corresponding to an environment configuration;

determining a functionality ~~of~~ for a unit of code;

determining ~~and~~ an environment configuration suitable for executing said unit of code; ~~and~~

~~automatically providing~~ generating a file name corresponding to said functionality for said unit of code;

searching said organizational structure to find a constituent member corresponding to said environment configuration that has been determined to be suitable for executing said unit of code; and

generating said unit of code if said file name is not found in said constituent member.

2. (Currently Amended) The method of claim 1, wherein said members of the organizational structure are directories, and said constitutent member corresponding to said environment configuration suitable for executing said unit of code is a constituent directory. ~~the method further comprising the step of locating a file having said file name in a directory corresponding to said environment configuration~~.

3. (Currently Amended) The method of claim 2, the method further comprising ~~the step of~~ naming said constituent directory to have a directory name corresponding to said environment configuration suitable for executing said unit of code.

4. (Currently Amended) The method of claim 2, wherein a plurality of characteristics of said environment configuration suitable for executing said unit of code include at least one of the group of a word size on a target processor, a word size on a host processor, an execution software type, an execution software version number and an operating system.

5. (Currently Amended) The method of claim 3, wherein said naming ~~step~~ determines said constituent directory name by the use of a checksum of a plurality of characteristics of said environment configuration suitable for executing said unit of code.

6. (Currently Amended) The method of claim 3, wherein said characteristics of said environment configuration suitable for executing said unit of code include at least one of the group of a word size on a target processor, a word size on a host processor, an execution software type, an execution software version number and an operating system.

7. (Previously Presented) The method of claim 1, wherein said file name is determined by the use of a checksum of characteristics of said functionality.

8. (Previously Presented) The method of claim 1, wherein said checksum is based on at least one of the group of a MD5 checksum and a CRC checksum.

9. (Previously Presented) The method of claim 7, wherein said characteristics of said functionality include at least one of the group of an input type and an output type of said unit of code, an operation on an input to said unit of code.

10. (Currently Amended) The method of claim 1, wherein said ~~step of automatically providing~~ generated a file name ~~provides said file name~~ also ~~corresponding~~ corresponds to said environment configuration suitable for executing said unit of code.

11. (Currently Amended) The method of claim 1, wherein said ~~step of automatically providing~~ generating a file name determines said file name by the use of a checksum of a plurality of characteristics of said functionality and said environment configuration suitable for executing said unit of code.

12. (Currently Amended) The method of claim 1, wherein said ~~step of automatically providing~~generating a file name determines said file name by the use of a consistent naming scheme representative of characteristics of said functionality.

13. (Previously Presented) The method of claim 1, wherein said file name is comprised of characters pertaining to an input type and an output type of said unit of code.

14. (Previously Presented) The method of claim 1, wherein said file name is comprised of characters pertaining to said functionality of said unit of code.

15. (Currently Amended) The method of claim 1, wherein said file name also corresponds to said environment configuration suitable for executing said unit of code.

16. (Previously Presented) The method of claim 1, wherein said unit of code is representative of a portion of a block diagram environment.

17. (Previously Presented) The method of claim 1, wherein said unit of code is representative of a portion of a modeling environment.

18. (Currently Amended) In an electronic device, a method for management of software, comprising ~~the steps of~~:

providing an organizational structure in a storage having a plurality of members, each member corresponding to an environment configuration;

determining a functionality ~~of~~ for a unit of code;

determining ~~and~~ an environment configuration suitable for executing said unit of code; ~~and~~

~~automatically providing~~ generating a function name corresponding to said functionality for said unit of code;~~.~~

searching the organizational structure to find a constituent member corresponding to said environment configuration that has been determined to be suitable for executing said unit of code; and

generating said unit of code if said function name is not found in said constituent member.

19. (Canceled)

20. (Currently Amended) The method of claim ~~19~~18, the method further comprising ~~the step of~~ naming said constituent member to have a ~~constituent~~ member name corresponding to said environment configuration suitable for executing said unit of code.

21. (Currently Amended) The method of claim 20, wherein a plurality of characteristics of said environment configuration suitable for executing said unit of code include at least one of the group of a word size on a target processor, a word size on a host processor, an execution software type, an execution software version number and an operating system.

22. (Currently Amended) The method of claim 20, wherein said naming ~~step~~ determines said ~~constituent~~ member name by the use of a checksum of a plurality of characteristics of said environment configuration suitable for executing said unit of code.

23. (Currently Amended) The method of claim 22, wherein said characteristics of said environment configuration suitable for executing said unit of code include at least one of the group of a word size on a target processor, a word size on a host processor, an execution software type, an execution software version number and an operating system.

24. (Currently Amended) The method of claim ~~19~~18, wherein said function name is determined by the use of a checksum of characteristics of said functionality.

25. (Currently Amended) The method of claim ~~19~~18, wherein said checksum is based on at least one of the group of a MD5 checksum and a CRC checksum.

26. (Previously Presented) The method of claim 24, wherein said characteristics of said functionality include at least one of the group of an input type and an output type of said unit of code, an operation on an input to said unit of code.

27. (Currently Amended) The method of claim ~~19~~18, wherein said ~~step of automatically providing~~ generated ~~a~~ function name ~~provides said function name~~ also ~~corresponding~~ corresponds to said environment configuration suitable for executing said unit of code.

28. (Currently Amended) The method of claim ~~19~~18, wherein said ~~step of automatically providing~~ generating a function name determines said function name by the use of a checksum of a plurality of characteristics of said functionality and said environment configuration suitable for executing said unit of code.

29. (Currently Amended) The method of claim ~~19~~18, wherein said ~~step of automatically providing~~ generating a function name determines said function name by the use of a consistent naming scheme representative of characteristics of said functionality.

30. (Currently Amended) The method of claim ~~19~~18, wherein said function name is comprised of characters pertaining to an input type and an output type of said unit of code.

31. (Currently Amended) The method of claim ~~19~~18, wherein said function name is comprised of characters pertaining to said functionality of said unit of code.

32. (Currently Amended) The method of claim ~~19~~18, wherein said function name also corresponds to said environment configuration suitable for executing said unit of code.

33. (Currently Amended) The method of claim ~~19~~18, wherein said unit of code is representative of a portion of a block diagram environment.

34. (Currently Amended) The method of claim ~~19~~18, wherein said unit of code is representative of a portion of a modeling environment.

35. (Currently Amended) In an electronic device, a method for management of software, comprising ~~the steps of~~:

providing an organizational structure in a storage having a plurality of members, each member corresponding to an environment configuration;

determining a functionality ~~of~~ for a unit of code;

determining a ~~and an~~ environment configuration suitable for executing said unit of code; ~~and~~

~~automatically providing~~ generating a macro name corresponding to said functionality for said unit of code~~.~~;

searching the organizational structure to find a constituent member corresponding to said environment configuration suitable that has been determined to be for executing said unit of code; and

generating said unit of code if said macro name is not found in said constituent member.

36. (Canceled)

37. (Currently Amended) The method of claim ~~36~~35, the method further comprising ~~the step of~~ naming said constituent member to have a ~~constituent~~ member name corresponding to said environment configuration suitable for executing said unit of code .

38. (Currently Amended) The method of claim 37, wherein a plurality of characteristics of said environment configuration suitable for executing said unit of code include at least one of the group of a word size on a target processor, a word size on a host processor, an execution software type, an execution software version number and an operating system.

39. (Currently Amended) The method of claim 37, wherein said naming ~~step~~ determines said ~~constituent~~ member name by the use of a checksum of a plurality of characteristics of said environment configuration suitable for executing said unit of code.

40. (Currently Amended) The method of claim 39, wherein said characteristics of said environment configuration suitable for executing said unit of code include at least one of the group of a word size on a target processor, a word size on a host processor, an execution software type, an execution software version number and an operating system.

41. (Previously Presented) The method of claim 37, wherein said macro name is determined by the use of a checksum of characteristics of said functionality.

42. (Previously Presented) The method of claim 37, wherein said checksum is based on at least one of the group of a MD5 checksum and a CRC checksum.

43. (Previously Presented) The method of claim 41, wherein said characteristics of said functionality include at least one of the group of an input type and an output type of said unit of code, an operation on an input to said unit of code.

44. (Currently Amended) The method of claim 37, wherein said ~~step of automatically providing~~ generated ~~a~~ macro name ~~provides said macro name~~ also ~~corresponding~~ corresponds to said environment configuration.

45. (Currently Amended) The method of claim 37, wherein said ~~step of automatically providing~~ generating a macro name determines said macro name by the use of a checksum of a plurality of characteristics of said functionality and said environment configuration suitable for executing said unit of code.

46. (Currently Amended) The method of claim 37, wherein said ~~step of automatically providing~~ generating a macro name determines said macro name by the use of a consistent naming scheme representative of characteristics of said functionality.

47. (Previously Presented) The method of claim 37, wherein said macro name is comprised of characters pertaining to an input type and an output type of said unit of code.

48. (Previously Presented) The method of claim 37, wherein said macro name is comprised of characters pertaining to said functionality of said unit of code.

49. (Currently Amended) The method of claim 37, wherein said macro name also corresponds to said environment configuration suitable for executing said unit of code.

50. (Previously Presented) The method of claim 37, wherein said unit of code is representative of a portion of a block diagram environment.

51. (Previously Presented) The method of claim 37, wherein said unit of code is representative of a portion of a modeling environment.

52. (Currently Amended) In an electronic device, a method for management of software, comprising ~~the steps of~~:

providing an organizational structure in a storage having a plurality of members, each member corresponding to an environment configuration;

determining a functionality ~~of~~ for a unit of code;

determining ~~and~~ an environment configuration suitable for executing said unit of code;

~~and~~

~~automatically providing~~generating a class name corresponding to said functionality for said unit of code;.

searching the organizational structure to find a constituent member corresponding to said environment configuration that has been determined to be suitable for executing said unit of code; and

generating said unit of code if said class name is not found in said constituent member.

53. (Canceled)

54. (Currently Amended) The method of claim ~~53~~52, the method further comprising ~~the step of~~ naming said constituent member to have a ~~constituent~~ member name corresponding to said environment configuration suitable for executing said unit of code.

55. (Currently Amended) The method of claim 54, wherein a plurality of characteristics of said environment configuration suitable for executing said unit of code include at least one of the group of a word size on a target processor, a word size on a host processor, an execution software type, an execution software version number and an operating system.

56. (Currently Amended) The method of claim 54, wherein said naming ~~step~~ determines said ~~constituent~~ member name by the use of a checksum of a plurality of characteristics of said environment configuration suitable for executing said unit of code.

57. (Currently Amended) The method of claim 56, wherein said characteristics of said environment configuration suitable for executing said unit of code include at least one of the

group of a word size on a target processor, a word size on a host processor, an execution software type, an execution software version number and an operating system.

58. (Previously Presented) The method of claim 52, wherein said class name is determined by the use of a checksum of characteristics of said functionality.

59. (Previously Presented) The method of claim 52, wherein said checksum is based on at least one of the group of a MD5 checksum and a CRC checksum.

60. (Previously Presented) The method of claim 58, wherein said characteristics of said functionality include at least one of the group of an input type and an output type of said unit of code, an operation on an input to said unit of code.

61. (Currently Amended) The method of claim 52, wherein said ~~step of automatically providing~~ generated ~~a~~ class name ~~provides said class name~~ also ~~corresponding~~ corresponds to said environment configuration.

62. (Currently Amended) The method of claim 52, wherein said ~~step of automatically providing~~ generating a class name determines said class name by the use of a checksum of a plurality of characteristics of said functionality and said environment configuration.

63. (Currently Amended) The method of claim 52, wherein said ~~step of automatically providing~~ generating a class name determines said class name by the use of a consistent naming scheme representative of characteristics of said functionality.

64. (Previously Presented) The method of claim 52, wherein said class name is comprised of characters pertaining to an input type and an output type of said unit of code.

65. (Previously Presented) The method of claim 52, wherein said class name is comprised of characters pertaining to said functionality of said unit of code.

66. (Currently Amended) The method of claim 52, wherein said class name also corresponds to said environment configuration suitable for executing said unit of code.

67. (Previously Presented) The method of claim 52, wherein said unit of code is representative of a portion of a block diagram environment.

68. (Previously Presented) The method of claim 52, wherein said unit of code is representative of a portion of a modeling environment.

69. (Currently Amended) In an electronic device, a method for management of software, comprising ~~the steps of~~:

providing an organizational structure in a storage having a plurality of ~~constituents~~ members, ~~said constituents~~ each member corresponding to ~~unique~~ an environment configuration~~s~~; ~~and~~

determining a functionality for a unit of code;

determining an environment configuration suitable for executing said unit of code;

providing an identifier corresponding to a functionality of a unit of code ~~in said organizational structure.~~;

searching the organizational structure to find a constituent member corresponding to said environment configuration that has been determined to be suitable for executing said unit of code; and

generating said unit of code if said identifier is not found in said constituent member.

70. (Canceled)

71. (Currently Amended) The method of claim ~~70~~69, the method further comprising ~~the step of~~ naming said constituent member to have a ~~constituent~~ member name corresponding to said environment configuration suitable for executing said unit of code.

72. (Currently Amended) The method of claim 71, wherein said naming ~~step~~ determines said ~~constituent~~ member name by the use of a checksum of a plurality of characteristics of said environment configuration suitable for executing said unit of code.

73. (Previously Presented) The method of claim 69, wherein said organizational structure is a directory structure and said identifier is a file name.

74. (Currently Amended) In an electronic device, a method for management of software, comprising ~~the steps of~~:

selecting a utility to process having a first functionality;

determining a characteristic of a first environment configuration suitable for operation of said utility;

searching an organizational structure ~~for~~ to find a constituent member corresponding to said first environment configuration;

creating said constituent member corresponding to said first environment configuration, if said constituent member corresponding to said first environment configuration is not found in said searching an organizational structure step;

generating~~e~~ a name for said first functionality;

searching said constituent member corresponding to said first environment configuration ~~for~~ to find an identifier of a unit of code, said identifier corresponding to said name for said first functionality; and

~~create~~ creating said unit of code having said first functionality and being suitable for execution in said first environment configuration, if said identifier is not found in said searching ~~said constituent corresponding to said first environment configuration step~~.

75. (Previously Presented) The method of claim 74, wherein said searching an organizational structure step uses a checksum of a plurality of characteristics of said first environment configuration.

76. (Currently Amended) The method of claim 74, wherein said ~~step of~~ searching a constituent member corresponding to said first environment configuration uses a checksum of a plurality of characteristics of said first functionality.

77. (Currently Amended) The method of claim 76, ~~the method~~ further comprising ~~the steps of~~:

generating a comment string corresponding to said first functionality of said unit of code; and

verifying said functionality of said unit of code by the use of said checksum and said comment string.

78. (Previously Presented) The method of claim 74, wherein said utility is operational in a block diagram environment.

79. (Previously Presented) The method of claim 74, wherein said utility is operational in a modeling environment.

80. (Previously Presented) The method of claim 74, wherein said identifier is a file name.

81. (Previously Presented) The method of claim 74, wherein said identifier is a function name.

82. (Previously Presented) The method of claim 74, wherein said identifier is a macro name.

83. (Previously Presented) The method of claim 74, wherein said identifier is a class name.

84. (Previously Presented) The method of claim 74, wherein said organizational structure is a directory structure.

85. (Previously Presented) The method of claim 74, wherein said organizational structure is a class structure.

86. (Currently Amended) The method of claim 74, wherein said constituent member is a file.

87. (Currently Amended) A computer readable medium having instructions stored therein which, when executed by a processor, cause the processor to:~~containing a software tool for executing a method for management of software, comprising the steps of:~~

select~~ing~~ a utility to process having a first functionality;

~~determining~~ determine a characteristic of a first environment configuration suitable for operation of said utility;

search~~ing~~ an organizational structure stored in a storage for a constituent member corresponding to said first environment configuration;

~~creating~~ create said constituent member corresponding to said first environment configuration, if said constituent member corresponding to said first environment configuration is not found in said searching an organizational structure step;

generat~~ing~~ a name for said first functionality;

search~~ing~~ said constituent member corresponding to said first environment configuration for an identifier of a unit of code, said identifier corresponding to said name for said first functionality; and

~~creating~~ create said unit of code having said first functionality and suitable for execution in said first environment configuration, if said file name is not found in said searching said constituent member corresponding to said first environment configuration step.

88. (Currently Amended) The medium of claim 87, wherein said searching an organizational structure ~~step~~ uses a checksum of a plurality of characteristics of said first environment configuration.

89. (Currently Amended) The medium of claim 87, wherein said ~~step of~~ searching a constituent member corresponding to said first environment configuration uses a checksum of a plurality of characteristics of said first functionality.

90. (Currently Amended) The medium of claim 89, the ~~method~~ instructions further ~~comprising~~causing said processor to ~~the steps of~~:

~~generating~~ generate a comment string corresponding to said first functionality of said unit of code; and

verify~~ing~~ said functionality of said unit of code by the use of said checksum and said comment string.

91. (Previously Presented) The medium of claim 87, wherein said utility is operational in a block diagram environment.

92. (Previously Presented) The medium of claim 87, wherein said utility is operational in a modeling environment.

93. (Previously Presented) The medium of claim 87, wherein said identifier is a file name.

94. (Previously Presented) The medium of claim 87, wherein said identifier is a function name.

95. (Previously Presented) The medium of claim 87, wherein said identifier is a macro name.

96. (Previously Presented) The medium of claim 87, wherein said identifier is a class name.

97. (Previously Presented) The medium of claim 87, wherein said organizational structure is a directory structure.

98. (Previously Presented) The medium of claim 87, wherein said organizational structure is a class structure.

99. (Currently Amended) The medium of claim 87, wherein said constituent member is a file.

100. (Currently Amended) In an electronic device, a system for managing code, comprising:

a storage for storing:

a functionality identifier, identifying a functionality of a unit of code;

an environment configuration identifier, identifying an environment configuration suitable for executing said unit of code; ~~and~~

a processor to:

execute a naming mechanism that derives a name for said unit of code corresponding to said functionality for said unit of code; and

locate a file having said name in a directory corresponding to said environment configuration.

101. (Canceled)

102. (Currently Amended) The system of claim ~~101~~100, wherein said system names said directory with a name corresponding to said environment configuration.

103. (Previously Presented) The system of claim 100, wherein said name comprises at least one of a file name, a function name, a macro name, a class name, and an identifier.

104. (Previously Presented) The system of claim 100, wherein a plurality of characteristics of said environment configuration include at least one of the group of a word size on a target processor, a word size on a host processor, an execution software type, an execution software version number and an operating system.

105. (Previously Presented) The system of claim 100, wherein said unit of code is representative of at least a portion of a block diagram environment.

106. (Previously Presented) The system of claim 100, wherein said unit of code is representative of at least a portion of a modeling environment.